Attorney Docket No.: 200310075-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

This is a U.S. Patent Application for:

Title:	ENHANCING IMAGE RESOLUTION
Inventor:	Mei Chen
Address:	284 Tyrella Ave., #11, Mountain View, CA 94043
Citizenship:	China

EXPRESS MAIL CERTIFICATE OF MAILING

EXPRESS MAIL NO.: ER212308409US

DATE OF DEPOSIT: April 15, 2004

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, PO Box 1450, Alexandria, VA 22313-1450.

Edouard Garcia
(Typed or printed name of person mailing paper or fee)

[illegible]
(Signature of person mailing paper or fee)

April 15, 2004
(Date signed)

ENHANCING IMAGE RESOLUTION

BACKGROUND

The resolution of an image is determined by the physical characteristics of the sensor that is used to capture the image. In some resolution enhancement approaches, the spatial dimension of an image is increased by interpolating between pixel values of a single image to generate an enhanced image. Both linear and non-linear interpolation techniques have been applied to a single image to produce an image with a higher spatial dimension than the original image. In these resolution enhancement techniques, however, the overall information content in the enhanced image is the same as the original image: the pixel number is increased, but the resolution is not improved. In other approaches, image resolution is increased by obtaining multiple displaced images of the scene and combining the information from the multiple images into an enhanced image. The process of reconstructing an image from several displaced images of the same scene often is referred to as "super resolution". In these resolution enhancement techniques, the overall information content in the enhanced image may be greater than the information content in the individual original images.

In some super resolution methods, one of the lower resolution images in a sequence of lower resolution images is designated as a reference image. The coordinate frame of the reference image is up-projected to the higher resolution level to define the coordinate frame of the higher resolution image. Motion vectors between the reference image and each of a selected number of lower-resolution images neighboring the reference image in the multi-image sequence are estimated. The motion vectors then are up-projected to the coordinate frame of the higher resolution image. The up-projected motion vectors are used to re-map the neighboring images to the coordinate frame of the higher resolution image. Next, the coordinate frame of the higher resolution image is populated with pixel values that are computed based on combinations of pixel values from the re-mapped lower resolution images. In some approaches, a non-linear interpolation technique is used to reconstruct the higher resolution image from the re-mapped pixel values.

SUMMARY

The invention features methods, machines, and machine-readable media for enhancing image resolution.

In one aspect of the invention, a respective motion map is computed for each pairing of a reference image and a respective image neighboring the reference image in a sequence of base images. Each motion map comprises a set of motion vectors mapping reference image pixels to respective neighboring image pixels. Respective regions of a target image are assigned to motion classes based on the computed motion maps. The target image has a target resolution level and the base images have a base resolution level equal to or lower than the target resolution level. Pixel values for the target image are computed based on corresponding pixel value contributions from the base images selected in accordance with the motion classes assigned to the target image regions.

Other features and advantages of the invention will become apparent from the following description, including the drawings and the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of an embodiment of a system for enhancing the resolution of a reference image in a sequence of base images.

FIG. 2A is a flow diagram of an embodiment of a method of enhancing the resolution of a reference image in a sequence of base images.

FIG. 2B is a flow diagram of an implementation of the embodiment of FIG. 2A.

FIG. 3 is a diagrammatic view of motion vectors mapping a pixel of a reference image to respective pixels of images neighboring the reference image in a sequence of base images.

FIGS. 4A-4D are diagrammatic views of different respective motions that typically appear in dynamic images.

FIG. 5A is a diagrammatic view of a neighboring image re-mapped to a coordinate frame of a subsequent reference image in accordance with a computed motion map.

FIG. 5B is a diagrammatic view of a neighboring image re-mapped to a coordinate frame of a preceding reference image in accordance with a computed motion map.

FIG. 6 shows an example of five windows that are used to compute motion maps in accordance with an implementation of the method of FIG. 2B.

FIG. 7 is a flow diagram of an embodiment of a method of assigning respective regions of a reference image to motion classes based on computed motion maps in accordance with an implementation of the method of FIG. 2B.

FIG. 8 is a diagrammatic view of segmented motion magnitude maps at different resolution levels in accordance with the method of FIG. 7.

FIG. 9 is a flow diagram of an embodiment of a method of computing pixel values for a target image based on pixel value contributions from a sequence of base images.

DETAILED DESCRIPTION

In the following description, like reference numbers are used to identify like elements. Furthermore, the drawings are intended to illustrate major features of exemplary embodiments in a diagrammatic manner. The drawings are not intended to depict every feature of actual embodiments nor relative dimensions of the depicted elements, and are not drawn to scale.

The image processing embodiments described below incorporate a dynamic approach to enhance the spatial resolution of an image sequence that allows different regions of the scene captured in the images to be treated differently. In this way, these embodiments are able to avoid artifacts that otherwise might result from treating all regions of the scene in the same way during the resolution enhancement process. In addition, these embodiments are able to dynamically tailor the image resolution enhance process in an intelligent way. In particular, these embodiments deploy image processing resources to different regions of an enhanced resolution image at varying computational intensity levels to achieve high quality resolution enhancement results in an accurate and efficient way.

FIG. 1 shows an embodiment of a system 10 for processing a sequence of base images 12 that includes a motion estimation module 14, a motion evaluation module 15, a motion segmentation module 16, an up-projection module 18, and an adaptive synthesis module 20. The system 10 is configured to produce a sequence of enhanced resolution target images 22. In some implementations, the system 10 also may be configured to produce a sequence of target images that have the same spatial resolution as the base images 12, but in which any compression, luminance and color aliasing artifacts associated with the base images are reduced.

In general, the modules 14-20 of system 10 are not limited to any particular hardware or software configuration, but rather they may be implemented in any computing or processing environment, including in digital electronic circuitry or in computer hardware, firmware, device driver, or software. For example, in some implementations, these modules 14-20 may be embedded in the hardware of any one of a wide variety of digital and analog electronic devices, including desktop and workstation computers, digital still image cameras, digital video cameras, printers, scanners, and portable electronic devices (e.g., mobile phones, laptop and notebook computers, and personal digital assistants).

The base image sequence 12 may correspond to an original base image sequence that was captured by an image sensor (e.g., a video image sequence or a still image sequence) or a processed version of such an original base image sequence. For example, the base image sequence 12 may consist of a sampling of the images selected from an original base image sequence that was captured by an image sensor or a compressed or reduced-resolution version of an original base image sequence that was captured by an image sensor. In order to achieve spatial resolution enhancement of the base image sequence 12, at least some of the base images correspond to displaced images of the same scene so that different samplings of the scene can be combined into an enhanced resolution target image.

Each target image 22 is produced from pixel value contributions from a selected set of the base images 12, including one that is designated the "reference image" and one or more base images that neighbor the reference image in the sequence. As used herein, the term "neighboring base images" refers to base images within a prescribed number of base images of each other in a base image sequence without regard to the temporal ordering of the neighboring base images in terms of capture time. In addition, the term "successive base images" refers to adjacent base images in a base image sequence that may be ordered chronologically or reverse-chronologically in terms of capture time. The number of neighboring images used to compute a target image, and the relative positions of the neighboring images in the sequence, are implementation-specific parameters. In some implementations, three successive neighboring base images on either side of the reference image in the base image sequence are processed with each reference image, for a total of seven base images that are processed for each target image.

Referring to FIG. 2A, in some embodiments, image processing system 10 processes base image sequence 12 to produce the target image sequence 22 as follows. Motion estimation module 14 computes a respective motion map for each pairing of a reference image and a respective image neighboring the reference image in the sequence of base images 12 (block 19). Each motion map comprises a set of motion vectors mapping reference image pixels to respective neighboring image pixels. Respective regions of a target image 22 are assigned to motion classes based on the computed motion maps (block 21). The target image has a target resolution level and the base images have a base resolution level equal to or lower than the target resolution level. Pixel values for the target image are computed based on corresponding pixel value contributions from the base images 12 selected in accordance with the motion classes assigned to the target image regions (block 23).

FIG. 2B shows one of many possible implementations of the image processing embodiment of FIG. 2A. Motion estimation module 14 computes a respective motion map (or motion correspondence map) for each pairing of the reference image and a respective neighboring image (block 24). Each motion map includes a set of motion vectors $\mathbf{u}_{i,k}$ that map each reference image pixel P_i to respective neighboring image pixels P_{i+1}, P_{i-1}, as shown in FIG. 3. The motion vectors estimate the inter-frame motion of features or objects appearing in the base images 12. In general, motion estimation module 14 may compute motion vectors based on any model for estimating the motion of image objects. For example, motion vectors may be computed based on an affine motion model that describes motions that typically appear in image sequences, including translation, rotation, zoom, and shear. Affine motion is parameterized by six parameters as follows:

$$U_x(x,y) = a_{x0} + a_{x1}x + a_{x2}y \qquad (1)$$

$$U_y(x,y) = a_{y0} + a_{y1}x + a_{y2}y \qquad (2)$$

wherein $U_x(x,y)$ and $U_y(x,y)$ are the x and y components of a velocity motion vector at point (x,y), respectively, and the a_k's are the affine motion parameters.

Examples of an affine motion model are illustrated in FIGS. 4A-4D. FIG. 4A shows parallel motion vectors that represent a translation of an object 26 at a constant distance from an image sensor (or image sensors). FIG. 4B shows vectors having a common focus of expansion that represent translation of object 26 in depth relative to the image sensor, or zooming (uniform scaling) motion. FIG. 4C shows concentric motion vectors that represent rotation of object 26 within the imaging plane. FIG. 4D represents rotation of object 26 with respect to Y axis.

In some embodiments, the motion maps of image pairs are represented as vector fields in the coordinate system of the reference image, which defines the coordinate system of the target image to be enhanced. A vector field *U(P)* the reference image $I_r(P)$, and the neighboring image $I_t(P)$ (e.g., one of the images preceding or succeeding the image to be enhanced in a image sequence), satisfy:

$$I_r(P) = I_t(P - U(P)) \qquad (3)$$

where $P = P(x,y)$ represents pixel coordinates. Therefore, each of the neighboring images can be warped to the coordinate frame of the corresponding reference image using equation (3) to create:

$$I_t^w(P) = I_t(P - U(P)) \qquad (4)$$

where $I_t^w(P)$ is the warped neighboring image. FIG. 5A shows an example of a preceding neighboring image re-mapped to the coordinate frame of a corresponding reference image in accordance with equation (4), where the cross-hatched areas are regions of the re-mapped image that do not overlap with the the coordinate frame of the reference image. Similarly, FIG. 5B shows an example of a subsequent neighboring image re-mapped to the coordinate frame of a corresponding reference image in accordance with equation (4).

In a typical image sequence of base images, if the motion vectors are computed correctly, the warped neighboring image should look very similar to the corresponding reference image. In the case of video sequences, the reference and neighboring images are captured at two different times. As a result, the pixel motion between images is due to both camera motion and also the motion of scene points moving independently. The motion of pixels is therefore unconstrained (non-parametric). Accordingly, in some embodiments, motion estimation module 14 computes movements of individual pixels or groups of pixels from a given base image to a neighboring base image based on a non-parametric optical flow model (or dense motion model). The motion estimates may be computed for one or both of forward and backwards transitions in time (i.e., from an earlier-captured base image to a later-captured base image, or from a later-captured base image to an earlier-captured base image).

In some embodiments, motion is modeled as a smoothly varying flow field, and motion analysis exploits local smoothness of optical flow fields. In this optical flow model, the movements are represented by velocity vectors (dx/dt, dy/dt) that describe how quickly a pixel (or a group of pixels) is moving across an image, and the direction of pixel movement. The optical flow model represents a projection of three-dimensional object motion onto the image sensor's two-dimensional image plane. Any one of a wide variety of optical flow computation methods may be used by the motion estimation module 14 to compute motion vectors. In some implementations, a multi-scale coarse-to-fine algorithm based on a gradient approach may be used to compute the optical flow motion vectors.

In some of these embodiments, the reference and neighboring image pairs are represented by Laplacian or Gaussian multi-resolution pyramids. In this way, these embodiments are able to accommodate a wide range of displacements, while avoiding excessive use of computational resources and generation of false matches. In particular, using a multi-resolution pyramid approach allows large displacements to be computed at low spatial resolution. Images at higher spatial resolution are used to improve the accuracy of displacement estimation by incrementally estimating finer displacements. Another advantage of using image pyramids is the reduction of false matches, which is caused mainly by the mismatches at higher resolutions under large motion. Motion estimation in a multi-resolution framework helps to eliminate problems of this type, since larger displacements are computed using images of lower spatial resolution, where they become small displacements due to sub-sampling.

In these embodiments, motion estimation module 14 uses a pyramid-based hierarchical image alignment technique to align two input images (i.e., a neighboring image and a corresponding reference image). A Laplacian or Gaussian pyramid is constructed from each of the two input images, and motion parameters are estimated in a coarse-to-fine manner. Within each pyramid level the sum of squared differences (SSD) measure integrated over regions of interest (which is initially the entire image region) is used as a match measure:

$$E(U(P)) = \sum_{P} (I_r(P) - I_t(P - U(P)))^2 \tag{5}$$

where I is the Laplacian or Gaussian filtered image intensity. The sum is computed over all the points P within the region and is used to denote the SSD error of the entire motion field within that region. The motion field is modeled by a set of global parameters (e.g., plane parameters) and local parameters (e.g., optical flow) as described above.

Numerical methods such as Gauss-Newton minimization is applied to the objective function described in equation (5) in order to estimate the unknown motion parameters and the resulting motion field. Starting with some initial values (typically zero), the hierarchical estimation algorithm iteratively refines the parameters in order to minimize the SSD error described in equation (5) from coarse to fine resolutions. After each motion estimation step, the current set of parameters is used to warp the neighboring image to the coordinate frame of the reference image, as described in equation (4), in order to reduce the residual displacement between the images.

The optical flow at each pixel is assumed to be locally constant in a small window around that pixel. The flow for a pixel is estimated by using all the pixels in its window. This process is repeated for each pixel and results in a smoothly varying flow field. In some implementations, dense optical flow is estimated using five windows 30, 32, 34, 36, 38, on and off-centered around each pixel under examination, as illustrated in FIG. 6. Local flow is computed for each window. The motion estimate that produces the smallest local error is used as the motion estimate for the pixel under consideration. Away from occlusion boundaries, the multiple windows 30-38 provide equally good estimates. However, at or near occlusion boundaries, the window with the best estimate will correspond only to the occluding surface. The non-optimal window estimates will come from the mixed estimate corresponding to the boundary between the occluding and occluded surfaces. Choosing the best estimate leads to crisp correspondence maps that are sub-pixel accurate at occluding boundaries.

Referring back to FIG. 2B, the motion evaluation module 15 computes an alignment accuracy map for each pairing of the reference image and a respective neighboring image based on the motion maps computed by the motion estimation module 14 (block 28). In some implementations, quality measures are computed based on correlations between the reference image and the re-mapped neighboring images. In some of these implementations, regions with low intensity variances optionally may be identified as being low-texture. The intensity means of corresponding low-texture regions in the aligned image pairs are compared. Pixels in low-texture regions with large difference in intensity means are assigned a correlation value of zero; whereas pixels in low-texture regions with little difference in intensity means are assigned a correlation value of one. The final alignment quality measure $\mathrm{M}_{\mathrm{Align}}$ is computed as follows,

$$\mathrm{M}_{Align} = \left[\sum_{P}\left(I_{ref}(P) - \overline{I_{ref}}\right)\left(I_{re-mapped}(P) - \overline{I_{remapped}}\right)\right] \Big/ \left[N\sigma_{ref}\sigma_{remapped}\right] \qquad (6)$$

when $$\sigma_{\mathrm{ref}}^{2} \leq \Omega \text{ AND } \sigma_{remapped}^{2} \leq \Omega$$

or $$\sigma_{\mathrm{N\_ref}}^{2} < \Omega_{\mathrm{N}} \text{ AND } \sigma_{N\_remapped}^{2} < \Omega_{N}$$

then $$\begin{cases} \mathrm{M}_{\mathrm{Align}} = 1.0, \text{if } \Delta\mu^{2} \leq \kappa\Omega \\ \mathrm{M}_{\mathrm{Align}} = 0.0, \text{else} \end{cases} \qquad (7)$$

where σ^2_{ref} and $\sigma^2_{remapped}$ are the respective reference and remapped neighboring image variances within the correlation window; $\sigma^2_N = \sigma^2 / (\mu^2 + c)$ is the mean normalized variance with μ being the mean and c a stabilizing constant to handle close-to-zero mean values; Ω, Ω_N and κ are thresholding parameters, and N is the number of pixels in the correlation window.

In some implementations, the total alignment quality is determined by computing the geometric mean of the quality measure for each of the color (e.g., Red, Green, and Blue) spectral bands of the base images 12. The alignment quality measures for each pairing of the reference image and a respective neighboring image are contained in respective alignment accuracy maps.

Referring back to FIG. 2B and to FIGS. 7 and 8, the motion segmentation module 16 assigns respective regions of the reference image to motion classes based on the computed motion maps (block 40). The motion segmentation module 16 computes motion magnitude maps from each motion map (block 42). In some implementations, motion magnitudes are computed by taking the square root of the sum of the squares of the x- and y-components of the motion vectors in the motion maps. The motion segmentation module 16 down-samples each of the computed motion magnitude maps 44 to a pyramid of coarser resolution levels, as shown in FIG. 8 (block 46).

The motion segmentation module 16 then segments pixels in the down-sampled motion magnitude maps 48 into motion classes (block 50). Motion segmentation module 16 may classify pixel regions in each down-sampled motion magnitude map into a respective set of motion classes (e.g., a high motion class region 58, intermediate motion class regions 54, 56, and a low motion class region 52) using any type of classification or segmentation method. For example, in some implementations, motion vectors in each motion magnitude map are segmented in accordance with a k-means clustering method. In these implementations, either the number of clusters or a set of clusters representing an initial partition between motion magnitudes in a given motion magnitude map may be pre-determined. The partition is refined iteratively by assigning pixels to each partition and re-computing the center of each cluster. The segmentation method iterates between the following steps:

1. Compute cluster centroids and use them as new cluster seeds; and
2. Assign each object to the nearest seed.

In some implementations, the final partition corresponds to a respective set of motion magnitude clusters 52, 54, 56, 58 for each coarse motion magnitude map in which the total distance between pixels (or pixel groups) and the centers of their respective clusters is minimized, while the distances between clusters are maximized.

The motion segmentation module 16 iteratively segments pixel motion magnitude maps from coarser resolution levels up to the original base image resolution level (block 60). The segmentation results from the previous resolution are used as the starting point for the same segmentation process applied to the next higher resolution level.

The separate motion class segmentation maps that are computed for each pairing of the reference image and a respective neighboring image are merged into a unified motion class segmentation map for the reference image. In some implementations, the motion segmentation module 16 assigns a given reference image pixel to the low motion class in the unified motion class segmentation map when the given pixel is assigned to the low motion class in all of the separate motion class segmentation maps. The motion segmentation module 16 assigns a given reference image pixel to the high motion class in the unified motion class segmentation map when the given pixel is assigned to the high motion class in any of the separate motion class segmentation maps. As explained above, some implementations include an intermediate motion class, where motion vectors that are assigned to the intermediate motion class have magnitudes higher than motion vectors assigned to the low motion class and lower than motion vectors assigned to the high motion class. In these implementations, the motion segmentation module 16 assigns a given reference image pixel to the intermediate motion class in the unified motion class segmentation map when the given pixel is unassigned to the high motion class in any of the separate motion class segmentation maps and is unassigned to the low motion class in all of the separate motion class segmentation maps.

Referring back to FIG. 2B, the up-projection module 18 up-projects the reference image, the motion class segmentation maps, the motion maps, and the alignment accuracy maps from the base image resolution level to the target image resolution level (block 62). In general, the reference image, the motion class segmentation maps, the motion maps, and the alignment accuracy maps may be up-projected to the target image resolution level using any type of resolution re-mapping or up-sampling technique. In some implementations, the reference image, the motion class segmentation maps, the motion maps, and the alignment accuracy maps are up-projected to a higher target image resolution using bi-cubic interpolation. In some other implementations, the reference image, the motion class segmentation maps, the motion maps, and the alignment accuracy maps are up-projected to a higher target image resolution by first using a 5-tap Gaussian filter to up-sample the reference image, the motion class segmentation maps, the motion maps, and the alignment accuracy maps, and then using interpolation to achieve sub-pixel accuracy.

Based on the up-projected motion maps, the adaptive synthesis module 20 re-maps the neighboring images to the coordinate frame of the target image (block 64). In some implementations, the neighboring images are up-projected to the target image resolution level and the up-projected neighboring images are re-mapped to the target image coordinate frame using the up-projected motion maps in accordance with application of equation (4) at the target image resolution level.

The adaptive synthesis module 20 applies a threshold to the up-projected alignment accuracy maps to produce respective synthesis maps for each of the neighboring images (block 66). The synthesis maps are used by the adaptive synthesis module 20 to classify motion vectors in each up-projected motion map into valid and invalid motion vector classes. In the illustrated embodiment, the threshold that is applied to the up-projected alignment accuracy maps is set to a level that ensures that the neighboring image pixels are sufficiently aligned with respect to the corresponding reference image pixels that they contain relevant information for reconstructing the target image. In some implementations, the alignment measures in the alignment accuracy maps are normalized to values in the range of ± 1 and the threshold is set to approximately 0.8. Pixels in the up-projected motion maps with alignment accuracy measures above the threshold are classified as valid motion pixels, whereas pixels with alignment accuracy measures below the threshold are classified as invalid motion pixels.

Referring to FIGS. 2 and 9, the adaptive synthesis module 20 computes values for the target image based on pixel value contributions from the reference image and a selected number of relevant neighboring images (block 68). In particular, the adaptive synthesis module 20 selects the pixel value contributions in accordance with the up-sampled motion class segmentation maps and the synthesis maps. In the illustrated embodiments, the number of relevant neighboring images decreases with the degree of the motion class to which the target image pixels are assigned. In some implementations, only the two nearest neighbors of the reference image contribute pixel values to high motion class target image pixels, only the four nearest neighbors (i.e., two nearest neighbors on either side) of the reference image contribute pixel values to intermediate motion class target image pixels, and only the six nearest neighbors (i.e., three nearest neighbors on either side) of the reference image contribute pixel values to low motion class target image pixels.

For pixels in the neighboring images that are identified as having invalid motion vectors according to the synthesis maps, their contribution to the target image is zero. For target image pixels whose corresponding neighboring image pixels are all identified as having invalid motion vectors (block 70), the adaptive synthesis module 20 computes target image pixel values by interpolating up-projected reference image pixel values (block 72). In some implementations, the adaptive synthesis module 20 uses bi-cubic interpolation to compute values of target image pixels associated with invalid motion vectors in all neighboring images.

For target image pixels assigned to the high motion class with at least one neighboring image pixel with valid motion vectors (blocks 70, 74), the adaptive synthesis module 20 computes pixel values by merging contributions from the reference image and the two relevant nearest re-mapped neighboring images (block 76). In some implementations, the high motion class target image pixels are computed by a pixel-wise weighted combination given by equation (8):

$$I_{Target} = \frac{\sum_j \omega_{T_j} \omega_{Align_j}(P) I_j(P)}{\sum_j \omega_{T_j} \omega_{Align_j}(P)} \tag{8}$$

where I_{Target} is the synthesized target image, ω_{T_j} is a weight for each of the re-mapped neighboring images, and ω_{Align_j} is a pixel-wise weight related to the alignment accuracy measure. In the case of a video stream, for example, weights ω_{T_j} may be set to values that are inversely proportional to the temporal distance between the reference image and the neighboring images. The weights ω_{Align_j} may be set to values that are proportional to the pixel-wise alignment quality measure if the measure is classified as a valid pixel in the corresponding synthesis map; otherwise it is set to be zero. The weights ω_{Align_j} may vary from pixel to pixel, and from image to image. This alignment-quality-related weighting guarantees that only relevant and valid information from well-aligned images is used during the rendering process and that unreliable information is ignored.

For target image pixels assigned to the intermediate and low motion classes (block 70, 74), the adaptive synthesis module 20 computes measures of local texture richness (block 78). Texture descriptors can be statistical, structural, or syntactic. In some implementations, a statistical descriptor is used. In these implementations, for a small local region around each pixel in the intermediate and low motion class regions, the adaptive synthesis module 20 computes the standard deviation of the Laplacian image, the skewness of the gradient value distribution, and the edge frequency as measures of local texture content. In some implementations, both the gradient image and the Laplacian image are computed during the motion estimation process (block 24; FIG. 2B). The adaptive synthesis module 20 may use any type of edge detection technique to find edges in the target and neighboring images. For example, in one implementation, the adaptive synthesis module 20 uses a Sobel edge detector, which performs a two-dimensional gradient measurement on the images, to compute edge directions and magnitudes. The Sobel edge detector uses a pair of 3x3 convolution masks, one of which estimates the gradient in the x-direction (columns) and the other of which estimates the gradient in the y-direction (rows).

The adaptive synthesis module 20 segments intermediate and low motion class pixels into high and low texture region classes based on the computed local texture richness measures (block 80). In some implementations, if the computed texture richness measure is below an empirically-determined threshold value, the adaptive synthesis module 20 segments the pixels into the low texture region class; otherwise the pixels are segmented into the high texture region class.

For intermediate and low motion class pixels assigned to the low texture region class in the reference image and all the corresponding neighboring images (block 82), the adaptive synthesis module 20 computes corresponding target image pixel values by interpolating up-projected reference image values (block 84). For intermediate and low motion class pixels assigned to the high texture region class in the reference image or any of the corresponding neighboring images (block 82), the adaptive synthesis module 20 computes target image pixel values by merging contributions from the up-projected reference image and the relevant re-mapped neighboring images in accordance with equation (9) (block 86):

$$I_{Target} = \frac{\sum_j \omega_{T_j} \omega_{Align_j} \omega_{Texture_j}(P) I_j(P)}{\sum_j \omega_{T_j} \omega_{Align_j} \omega_{Texture_j}(P)} \tag{9}$$

where $\omega_{Texture_j}$ is a weight with values ranging from 0 to 1 that are set based on the computed local texture richness measures. For example, in some implementations, $\omega_{Texture_j}$ corresponds to the computed local texture measure normalized to the 0 to 1 value range.

The above-described resolution enhancement embodiments may be applied to one or all of the luminance and chrominance components of the base images 12. In some embodiments, the resulting resolution-enhanced target images 22 may be subjected to one or more post-processing methods, including color re-mapping, sharpening, and de-scintillation methods.

Other embodiments are within the scope of the claims.

The systems and methods described herein are not limited to any particular hardware or software configuration, but rather they may be implemented in any computing or processing environment, including in digital electronic circuitry or in computer hardware, firmware, or software. In general, the systems may be implemented, in part, in a computer process product tangibly embodied in a machine-readable storage device for execution by a computer processor. In some embodiments, these systems preferably are implemented in a high level procedural or object oriented processing language; however, the algorithms may be implemented in assembly or machine language, if desired. In any case, the processing language may be a compiled or interpreted language. The methods described herein may be performed by a computer processor executing instructions organized, for example, into process modules to carry out these methods by operating on input data and generating output. Suitable processors include, for example, both general and special purpose microprocessors. Generally, a processor receives instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer process instructions include all forms of non-volatile memory, including, for example, semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM. Any of the foregoing technologies may be supplemented by or incorporated in specially designed ASICs (application-specific integrated circuits).